IEEE Xplore® RELEASE 2.0

Welcome United States Patent and Trademark Office

Search Session History

Wed, 22 Jun 2005, 12:18:22 PM EST

Edit an existing query or compose a new query in the Search Query Display.

Search Query Display

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Recent Search Queries		Results
#1	(hilton r.<in>au)	5
#2	(hilton r.<in>au)	5
#3	(ronald hilton<in>metadata)	0
#4	(instruction set simulator<in>metadata)	56
#5	(instruction set simulation<in>metadata)	14
#6	(emulation<in>metadata) <and> (instruction<in>metadata) &... legacy	1
#7	(self modifying code<in>metadata) <and> (emulation<in>meta...	0
#8	(instruction<in>metadata) <and> (translat<in>metadata)	0
#9	(instruction<in>metadata) <and> (translation<in>metadata)...	134
#10	((instruction<in>metadata) <and> (translation<in>metadata)... emulation	5

Terms used dynamic binary translation Found 27,371 of 157,956

Sort results by relevance | Save results to a Binder | Try an Advanced Search
Display results expanded form | Search Tips | Try this search in The ACM Guide
Open results in a new window

Results 41 - 60 of 200 Result page: previous 1 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

41 An instruction set and microarchitecture for instruction level distributed processing
Ho-Seop Kim, James E. Smith
May 2002 **ACM SIGARCH Computer Architecture News**, Volume 30 Issue 2
Full text available: pdf(1.06 MB) Publisher Site Additional information: full citation, abstract, citations, citings, index terms

An instruction set architecture (ISA) suitable for future microprocessor design constraints is proposed. The ISA has hierarchical register files with a small number of accumulators at the top. The instruction stream is divided into chains of dependent instructions (strands) where intra-strand dependences are passed through the accumulator. The general-purpose register file is used for communication between strands and for holding global values that have many consumers.A microarchitecture to supp ...

42 Randomized instruction set emulation
Elena Gabriela Barrantes, David H. Ackley, Stephanie Forrest, Darko Stefanović
February 2005 **ACM Transactions on Information and System Security (TISSEC)**, Volume 8 Issue 1
Full text available: pdf(374.44 KB) Additional information: full citation, abstract, references, index terms

Injecting binary code into a running program is a common form of attack. Most defenses employ a "guard the doors" approach, blocking known mechanisms of code injection. *Randomized instruction set emulation* (RISE) is a complementary method of defense, one that performs a hidden randomization of an application's machine code. If foreign binary code is injected into a program running under RISE, it will not be executable because it will not know the proper randomization. The pape ...

Keywords: Automated diversity, randomized instruction sets, software diversity

43 Hardware supported optimization: Static strands: safely collapsing dependence chains for increasing embedded power efficiency
Peter G. Sassone, D. Scott Wills, Gabriel H. Loh
June 2005 **Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems**
Full text available: pdf(274.43 KB) Additional information: full citation, abstract, references, index terms

Modern embedded processors are designed to maximize execution efficiency--the amount of performance achieved per unit of energy dissipated while meeting minimum performance levels. To increase this efficiency we propose utilizing *static strands*, dependence chains without fan-out which are exposed by a compiler pass. These dependent instructions are resequenced to be sequential and annotated to communicate their location to the hardware. Importantly, this modified application is binary com ...

Keywords: architecture, dependency collapsing, embedded, energy, sequentiality

44 tcc: a system for fast, flexible, and high-level dynamic code generation
Massimiliano Poletto, Dawson R. Engler, M. Frans Kaashoek
May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation**, Volume 32 Issue 5
Full text available: pdf(1.96 MB) Additional information: full citation, abstract, references, citings, index terms

tcc is a compiler that provides efficient and high-level access to dynamic code generation. It implements the 'C ("Tick-C") programming language, an extension of ANSI C that supports dynamic code generation [15]. 'C gives power and flexibility in specifying dynamically generated code: whereas most other

systems use annotations to denote run-time invariants. 'C allows the programmer to specify and compose arbitrary expressions and statements at run time. This degree of control is needed to effici ...

45 TraceBack: first fault diagnosis by reconstruction of distributed control flow
Andrew Ayers, Richard Schooler, Chris Metcalf, Anant Agarwal, Junghwan Rhee, Emmett Witchel
May 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation**, Volume 40 Issue 6
Full text available: pdf(347.77 KB) Additional information: full citation, abstract, references, index terms

Faults that occur in production systems are the most important faults to fix, but most production systems lack the debugging facilities present in development environments. TraceBack provides debugging information for production systems by providing execution history data about program problems (such as crashes, hangs, and exceptions). TraceBack supports features commonly found in production environments such as multiple threads, dynamically loaded modules, multiple source languages (e.g., Java ...

Keywords: fault diagnosis, instrumentation

46 Compilation and run-time systems: Vacuum packing: extracting hardware-detected program phases for post-link optimization
Ronald D. Barnes, Erik M. Nystrom, Matthew C. Merten, Wen-mei W. Hwu
November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.24 MB) Publisher Site Additional information: full citation, citations, references, citings, index terms

This paper presents Vacuum Packing, a new approach to profile-based program optimization. Instead of using traditional aggregate or summarized execution profile weights, this approach uses a transparent hardware profiler to automatically detect execution phases and record branch profile information for each new phase. The code extraction algorithm then produces code packages that are specially formed for their corresponding phases. The algorithm compensates for the incomplete and often incoheren ...

47 The Performance of Runtime Data Cache Prefetching in a Dynamic Optimization System
Jiwei Lu, Howard Chen, Rao Fu, Wei-Chung Hsu, Bobbie Othmer, Pen-Chung Yew, Dong-Yuan Chen
December 2003 **Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(455.70 KB) Additional information: full citation, abstract, citings, index terms

Traditional software controlled data cache prefetching isoften ineffective due to the lack of runtime cache miss andmiss address information. To overcome this limitation, weimplement runtime data cache prefetching in the dynamicoptimization system ADORE (ADaptive Object code RE-optimization).Its performance has been compared withstatic software prefetching on the SPEC2000 benchmarksuite. Runtime cache prefetching shows better performance.On an Itanium 2 based Linux workstation, it can increasepe ...

48 Supporting autonomic computing functionality via dynamic operating system kernel aspects
Michael Engel, Bernd Freisleben
March 2005 **Proceedings of the 4th international conference on Aspect-oriented software development**
Full text available: pdf(329.12 KB) Additional information: full citation, abstract, references, index terms

To master the complexity of software systems in the presence of unexpected events potentially affecting system operation, the *Autonomic Computing Initiative* [16] aims to build systems that have the ability to control and organize themselves to meet unforeseen changes in the hard- and software environment.The basic principles employed by autonomic computing are self-configuration, self-optimization, self-healing and self-protection. Typically, these principles are cross-cutting concerns, s ...

Keywords: NetBSD, autonomic computing, dynamic aspects, operating system kernel, organic computing

49 Dynamo: a transparent dynamic optimization system
Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(458.55 KB) Additional information: full citation, abstract, references, citings, index terms

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

50 Slim binaries
Michael Franz, Thomas Kistler
December 1997 **Communications of the ACM**, Volume 40 Issue 12
Full text available: pdf(150.83 KB) Additional Information: full citation, references, citings, index terms

51 Using the SimOS machine simulator to study complex computer systems
Mendel Rosenblum, Edouard Bugnion, Scott Devine, Stephen A. Herrod
January 1997 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 7 Issue 1
Full text available: pdf(731.78 KB) Additional Information: full citation, references, citings, index terms, review

Keywords: computer architecture, computer simulation, computer system performance analysis, operating systems

52 Overcoming the challenges to feedback-directed optimization (Keynote Talk)
Michael D. Smith
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7
Full text available: pdf(1.32 MB) Additional Information: full citation, abstract, references, citings, index terms

Feedback-directed optimization (FDO) is a general term used to describe any technique that alters a program's execution based on tendencies observed in its present or past runs. This paper reviews the current state of affairs in FDO and discusses the challenges inhibiting further acceptance of these techniques. It also argues that current trends in hardware and software technology have resulted in an execution environment where immutable executables and traditional static optimizations are ...

53 Binary translation
Richard L. Sites, Anton Chernoff, Matthew B. Kirk, Maurice P. Marks, Scott G. Robinson
February 1993 **Communications of the ACM**, Volume 36 Issue 2
Full text available: pdf(4.64 MB) Additional Information: full citation, references, citings, index terms

Keywords: CISC computers, RISC computers, binary translation, computer architecture, processor architecture translation

54 Exploring Code Cache Eviction Granularities in Dynamic Optimization Systems
Kim Hazelwood, James E. Smith
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(758.37 KB) Additional Information: full citation, abstract, index terms

Dynamic optimization systems store optimized or translatedcode in a software-managed code cache in order tomaximize reuse of transformed code. Code caches storesuperblocks that are not fixed in size, may contain linksto other superblocks, and carry a high replacement overhead.These additional constraints reduce the effectivenessof conventional hardware-based cache management policies.In this paper, we explore code cache managementpolicies that evict large blocks of code from the code cache,thus ...

55 DCG: an efficient, retargetable dynamic code generation system
Dawson R. Engler, Todd A. Proebsting
November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29 , 28 Issue 11 , 5
Full text available: pdf(878.79 KB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic code generation allows aggressive optimization through the use of runtime information. Previous systems typically relied on ad hoc code generators that were not designed for retargetability, and did not shield the client from machine-specific details. We present a system, dcg, that allows clients to specify dynamically generated code in a machine-independent manner. Our one-pass code generator is easily retargeted and extremely efficient (code generation costs appro ...

56 Instruction path coprocessors
Yuan Chou, John Paul Shen
May 2000 **ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture**, Volume 28 Issue 2

Full text available: pdf(134.64 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the concept of an Instruction Path Coprocessor (I-COP), which is a programmable on-chip coprocessor, with its own mini-instruction set, that operates on the core processor's instructions to transform them into an internal format that can be more efficiently executed. It is located off the critical path of the core processor to ensure that it does not negatively impact the core processor's cycle time or pipeline depth. An I-COP is highly versatile and can be used ...

57 DISE: a programmable macro engine for customizing applications
Marc L. Corliss, E. Christopher Lewis, Amir Roth
May 2003 **ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture**, Volume 31 Issue 2
Full text available: pdf(335.30 KB) Additional Information: full citation, citings, references, index

Dynamic Instruction Stream Editing (DISE) is a cooperative software-hardware scheme for efficiently adding customization functionality---e.g, safety/security checking, profiling, dynamic code decompression, and dynamic optimization---to an application. In DISE, application customization functions (ACFs) are formulated as rules for macro-expanding certain instructions into parameterized instruction sequences. The processor executes the rules on the fetched instructions, feeding the executi ...

58 Adaptive code unloading for resource-constrained JVMs
Lingli Zhang, Chandra Krintz
June 2004 **ACM SIGPLAN Notices , Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems**, Volume 39 Issue 7
Full text available: pdf(304.78 KB) Additional Information: full citation, abstract, references, cover terms

Compile-only JVMs for resource-constrained embedded systems have the potential for using device resources more efficiently than interpreter-only systems since compilers can produce significantly higher quality code and code can be stored and reused for future invocations. However, this additional storage requirement for reuse of native code bodies, introduces memory overhead not imposed in interpreter-based systems.In this paper, we present a Java Virtual Machine (JVM) extension for adaptive cod ...

Keywords: JIT, JVM, code unloading, code-size reduction, resource-constrained devices

59 Identifying and Exploiting Spatial Regularity in Data Memory References
Tushar Mohan, Bronis R. de Supinski, Sally A. McKee, Frank Mueller, Andy Yoo, Martin Schulz
November 2003 **Proceedings of the 2003 ACM/IEEE conference on Supercomputing**
Full text available: pdf(264.76 KB) Additional Information: full citation, abstract

The growing processor/memory performance gap causes the performance of many codes to be limited by memory accesses. If known to exist in an application, strided memory accesses forming streams can be targeted by optimizations such as prefetching, relocation, remapping, and vector loads. Undetected, they can be a significant source of memory stalls in loops. Existing stream-detection mechanisms either require special hardware, which may not gather statistics for subsequent analysis, or are limite ...

60 Contributed papers: The donkey strikes back: extending the dynamic interpretation "constructively"
Tim Fernando
April 1993 **Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics**
Full text available: pdf(619.83 KB) PostScript File Additional Information: full citation, abstract, references

The dynamic interpretation of a formula as a binary relation (inducing transitions) on states is extended by alternative treatments of implication, universal quantification, negation and disjunction that are more "dynamic" (in a precise sense) than the usual reductions to tests from quantified dynamic logic (which, nonetheless, can be recovered from the new connectives). An analysis of the "donkey" sentence followed by the assertion "It will kick back" is provided.

Results 41 - 60 of 200 Result page: previous 1 2 3 4 5 6 7 8 9 10 next

PORTAL USPTO

Search: The ACM Digital Library The Guide

+instruction +set translation simulation emulation SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used instruction set translation simulation emulation Found 27,448 of 157,956

Sort results by relevance Save results to a Binder Try an Advanced Search

Display results expanded form Search Tips Try this search in The ACM Guide

Open results in a new window

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Cycle Accurate Binary Translation for Simulation Acceleration in Rapid Prototyping of SoCs
Jurgen Schnerr, Oliver Bringmann, Wolfgang Rosenstiel
March 2005 **Proceedings of the conference on Design, Automation and Test in Europe - Volume 2**
Full text available: pdf(135.92 KB) Additional Information: full citation, abstract

In this paper, the application of a cycle accurate binary translator for rapid prototyping of SoCs will be presented. This translator generates code to run on a rapid prototyping system consisting of a VLIW processor and FPGAs. The generated code is annotated with information that triggers cycle generation for the hardware in parallel to the execution of the translated program. The VLIW processor executes the translated program whereas the FPGAs contain the hardware for the parallel cycle genera ...

2 Trace-driven memory simulation: a survey
Richard A. Uhlig, Trevor N. Mudge
June 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 2
Full text available: pdf(536.11 KB) Additional Information: full citation, abstract, references, citings, index terms, review

As the gap between processor and memory speeds continues to widen, methods for evaluating memory system designs before they are implemented in hardware are becoming increasingly important. One such method, trace-driven memory simulation, has been the subject of intense interest among researchers and has, as a result, enjoyed rapid development and substantial improvements during the past decade. This article surveys and analyzes these developments by establishing criteria for evaluating trac ...

Keywords: TLBs, caches, memory management, memory simulation, trace-driven simulation

3 Randomized instruction set emulation
Elena Gabriela Barrantes, David H. Ackley, Stephanie Forrest, Darko Stefanović
February 2005 **ACM Transactions on Information and System Security (TISSEC)**, Volume 8 Issue 1
Full text available: pdf(324.44 KB) Additional Information: full citation, abstract, references, index terms

Injecting binary code into a running program is a common form of attack. Most defenses employ a "guard the doors" approach, blocking known mechanisms of code injection. *Randomized instruction set emulation* (RISE) is a complementary method of defense, one that performs a hidden randomization of an application's machine code. If foreign binary code is injected into a program running under RISE, it will not be executable because it will not know the proper randomization. The pape ...

Keywords: Automated diversity, randomized instruction sets, software diversity

4 Shade: a fast instruction-set simulator for execution profiling
Bob Cmelik, David Keppel
May 1994 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems**, Volume 22 Issue 1
Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references, citings, index terms

Tracing tools are used widely to help analyze, design, and tune both hardware and software systems. This paper describes a tool called Shade which combines efficient instruction-set simulation with a flexible, extensible trace generation capability. Efficiency is achieved by dynamically compiling and caching code to simulate and trace the application program. The user may control the extent of tracing in a variety of ways; arbitrarily detailed application state information may be collected ...

5 RABIT: A New Framework for Runtime Emulation and Binary Translation
Suranjan Pramanik, Shambhu J. Upadhyaya
April 2004 **Proceedings of the 37th annual symposium on Simulation**
Full text available: pdf(232.93 KB) Additional Information: full citation, abstract

Binary translation has been widely used as object codemigration across different architectures. Most currentworks are targeted towards running an existing (old architecture)binary version of a complex application on a newerarchitecture and so availability of resources is not a problem.In this paper we propose a technique called RABIT foremulating a newer architecture's binary on an older one.RABIT will allow the consumers to emulate the applicationsdeveloped for newer hardware on their older mac ...

6 Efficient instruction cache simulation and execution profiling with a threaded-code interpreter
Peter S. Magnusson
December 1997 **Proceedings of the 29th conference on Winter simulation**
Full text available: pdf(912.22 KB) Additional Information: full citation, references, index terms

7 Performance of the VAX-11/780 translation buffer: simulation and measurement
Douglas W. Clark, Joel S. Emer
February 1985 **ACM Transactions on Computer Systems (TOCS)**, Volume 3 Issue 1
Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, references, citings, index terms

A virtual-address translation buffer (TB) is a hardware cache of recently used virtual-to-physical address mappings. The authors present the results of a set of measurements and simulations of translation buffer performance in the VAX-11/780. Two different hardware monitors were attached to VAX-11/780 computers, and translation buffer behavior was measured. Measurements were made under normal time-sharing use and while running reproducible synthetic time-sharing work loads. Reported measure ...

8 Microarchitecture-level power analysis and optimization techniques: Hybrid simulation for embedded software energy estimation
Anish Muttreja, Anand Raghunathan, Srivaths Ravi, Niraj K. Jha
June 2005 **Proceedings of the 42nd annual conference on Design automation**
Full text available: pdf(151.51 KB) Additional Information: full citation, abstract, references, index terms

Software energy estimation is a critical step in the design of energy-efficient embedded systems. Instruction-level simulation techniques, despite several advances, remain too slow for iterative use in system-level exploration. In this paper, we propose a methodology called *hybrid simulation*, which combines instruction set simulation with selective native execution (execution of some parts of the program directly on the simulation host computer), thereby overcoming the disadvantages of in ...

Keywords: embedded software, energy estimation, energy macromodels, hybrid simulation, pointers analysis

9 Dynamic translation: Retargetable and reconfigurable software dynamic translation
K. Scott, N. Kumar, S. Velusamy, B. Childers, J. W. Davidson, M. L. Soffa
March 2003 **Proceedings of the International symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.14 MB) Additional Information: full citation, abstract, references, citings, index terms

Software dynamic translation (SDT) is a technology that permits the modification of an executing program's instructions. In recent years, SDT has received increased attention, from both industry and academia, as a feasible and effective approach to solving a variety of significant problems. Despite this increased attention, the task of initiating a new project in software dynamic translation remains a difficult one. To address this concern, and in particular, to promote the adoption of SDT techn ...

10 A dynamic binary translation approach to architectural simulation
Harold W. Cain, Kevin M. Lepak, Mikko H. Lipasti
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(89.63 KB) Additional Information: full citation, abstract, index terms

We present the design of a PowerPC-based simulation infrastructure for architectural research. Our infrastructure uses an execution-driven out-of-order processor timing simulator from the SimpleScalar tool set. While porting SimpleScalar to the PowerPC architecture, we would like to remain compatible with other versions of SimpleScalar. We accomplish this by performing dynamic binary translation of the PowerPC instruction set architecture to the SimpleScalar instruction set architecture, and by ...

[11] Instruction fetching: coping with code bloat
Richard Uhlig, David Nagle, Trevor Mudge, Stuart Sechrest, Joel Emer
May 1995 **ACM SIGARCH Computer Architecture News , Proceedings of the 22nd annual international symposium on Computer architecture**, Volume 23 Issue 2
Full text available: pdf(1.47 MB) Additional information: full citation, abstract, references, citings, index terms

Previous research has shown that the SPEC benchmarks achieve low miss ratios in relatively small instruction caches. This paper presents evidence that current software-development practices produce applications that exhibit substantially higher instruction-cache miss ratios than do the SPEC benchmarks. To represent these trends, we have assembled a collection of applications, called the Instruction Benchmark Suite (IBS), that provides a better test of instruction-cache performance. We discuss th ...

[12] Implementation aspects of a SPARC V9 complete machine simulator
Bill Clarke, Adam Czezowski, Peter Strazdins
January 2002 **Australian Computer Science Communications , Proceedings of the twenty-fifth Australasian conference on Computer science - Volume 4**, Volume 24 Issue 1
Full text available: pdf(1.33 MB) Additional information: full citation, abstract, references, index terms

In this paper we present work in progress in the development of a complete machine simulator for the UltraSPARC, an implementation of the SPARC V9 architecture. The complexity of the UltraSPARC ISA presents many challenges in developing a reliable and yet reasonably efficient implementation of such a simulator. Our implementation includes a heavily object-oriented design for the simulator modules and infrastructure, caching of repeated computations for performance, adding an OS (system call) emu ...

Keywords: SMP, SPARC V9 ISA, UltraSPARC, complete machine simulator, execution-driven simulation, object-oriented design

[13] A retargetable, ultra-fast instruction set simulator
Jianwen Zhu, Daniel D. Gajski
January 1999 **Proceedings of the conference on Design, automation and test in Europe**
Full text available: pdf(57.15 KB) Additional information: full citation, citings, index terms

[14] Machine-adaptable dynamic binary translation
David Ung, Cristina Cifuentes
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7
Full text available: pdf(1.25 MB) Additional information: full citation, abstract, references, citings, index terms

Dynamic binary translation is the process of translating and optimizing executable code for one machine to another at runtime, while the program is "executing" on the target machine.

Dynamic translation techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translations, by providing a framework based on specifications of machines that ...

Keywords: binary translation, dynamic compilation, dynamic execution, emulation, interpretation

[15] Using the SimOS machine simulator to study complex computer systems
Mendel Rosenblum, Edouard Bugnion, Scott Devine, Stephen A. Herrod
January 1997 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 7 Issue 1
Full text available: pdf(741.78 KB) Additional information: full citation, references, citings, index terms, review

Keywords: computer architecture, computer simulation, computer system performance analysis, operating systems

[16] Dynamic translation: Dynamic binary translation for accumulator-oriented architectures
Ho-Seop Kim, James E. Smith
March 2003 **Proceedings of the International symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.12 MB) Publisher Site Additional information: full citation, abstract, references, citings, index terms

A dynamic binary translation system for a co-designed virtual machine is described and evaluated. The underlying hardware directly executes an accumulator-oriented instruction set that exposes instruction dependence chains (strands) to a distributed microarchitecture containing a simple instruction pipeline. To support conventional program binaries, a source instruction set (Alpha in our study) is dynamically translated to the target accumulator instruction set. The binary translator identifies ...

[17] An instruction set and microarchitecture for instruction level distributed processing
Ho-Seop Kim, James E. Smith
May 2002 **ACM SIGARCH Computer Architecture News**, Volume 30 Issue 2
Full text available: pdf(1.08 MB) Publisher Site Additional information: full citation, abstract, references, citings, index terms

An instruction set architecture (ISA) suitable for future microprocessor design constraints is proposed. The ISA has hierarchical register files with a small number of accumulators at the top. The instruction stream is divided into chains of dependent instructions (strands) where intra-strand dependences are passed through the accumulator. The general-purpose register file is used for communication between strands and for holding global values that have many consumers.A microarchitecture to supp ...

[18] Intrusion detection: Randomized instruction set emulation to disrupt binary code injection attacks
Elena Gabriela Barrantes, David H. Ackley, Trek S. Palmer, Darko Stefanovic, Dino Dai Zovi
October 2003 **Proceedings of the 10th ACM conference on Computer and communications security**
Full text available: pdf(100.71 KB) Additional information: full citation, abstract, references, citings, index terms

Binary code injection into an executing program is a common form of attack. Most current defenses against this form of attack use a 'guard all doors' strategy, trying to block the avenues by which execution can be diverted. We describe a complementary method of protection, which disrupts foreign code execution regardless of how the code is injected. A unique and private machine instruction set for each executing program would make it difficult for an outsider to design binary attack code against ...

Keywords: automated diversity, emulation, information hiding, language randomization, obfuscation, security

[19] Emulation of large systems
S. G. Tucker
December 1965 **Communications of the ACM**, Volume 8 Issue 12
Full text available: pdf(1.20 MB) Additional information: full citation, citings, index terms

[20] Binary translation and architecture convergence issues for IBM system/390
Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye
May 2000 **Proceedings of the 14th international conference on Supercomputing**
Full text available: pdf(1.44 MB) Additional information: full citation, abstract, references, index terms

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next

Terms used dynamic translation Found 48,817 of 157,936

Sort results by relevance | Save results to a Binder | Try an Advanced Search
Display results expanded form | Search Tips | Try this search in The ACM Guide
Open results in a new window

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Dynamic translation: Retargetable and reconfigurable software dynamic translation
K. Scott, N. Kumar, S. Velusamy, B. Childers, J. W. Davidson, M. L. Soffa
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.16 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Software dynamic translation (SDT) is a technology that permits the modification of an executing program's instructions. In recent years, SDT has received increased attention, from both industry and academia, as a feasible and effective approach to solving a variety of significant problems. Despite this increased attention, the task of initiating a new project in software dynamic translation remains a difficult one. To address this concern, and in particular, to promote the adoption of SDT techn ...

2 Dynamic translation: Dynamic binary translation for accumulator-oriented architectures
Ho-Seop Kim, James E. Smith
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.11 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

A dynamic binary translation system for a co-designed virtual machine is described and evaluated. The underlying hardware directly executes an accumulator-oriented instruction set that exposes instruction dependence chains (strands) to a distributed microarchitecture containing a simple instruction pipeline. To support conventional program binaries, a source instruction set (Alpha in our study) is dynamically translated to the target accumulator instruction set. The binary translator identifies ...

3 Machine-adaptable dynamic binary translation
David Ung, Cristina Cifuentes
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7
Full text available: pdf(1.22 MB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic binary translation is the process of translating and optimizing executable code for one machine to another at runtime, while the program is "executing" on the target machine.

Dynamic translation techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translations, by providing a framework based on specifications of machines that ...

Keywords: binary translation, dynamic compilation, dynamic execution, emulation, interpretation

4 Using Dynamic Binary Translation to Fuse Dependent Instructions
Shiliang Hu, James E. Smith
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(240.50 KB) Additional Information: full citation, abstract, index terms

Instruction scheduling hardware can be simplifiedand easily pipelined if pairs of dependent instructionsare fused so they share a single instruction schedulingslot. We study an implementation of the x86 ISA thatdynamically translates x86 code to an underlying ISAthat supports instruction fusing. A

microarchitecturethat is co-designed with the fused instruction set completesthe implementation.In this paper, we focus on the dynamic binarytranslator for such a co-designed x86 virtual machine.The dy ...

5 Dynamic translation: The Transmeta Code Morphing™ Software: using speculation, recovery, and adaptive retranslation to address real-life challenges
James C. Dehnert, Brian K. Grant, John P. Banning, Richard Johnson, Thomas Kistler, Alexander Klaiber, Jim Mattson
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.25 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms, review

Transmeta's Crusoe microprocessor is a full, system-level implementation of the x86 architecture, comprising a native VLIW microprocessor with a software layer, the **Code Morphing Software** (CMS), that combines an interpreter, dynamic binary translator, optimizer, and runtime system. In its general structure, CMS resembles other binary translation systems described in the literature, but it is unique in several respects. The wide range of PC workloads that CMS must handle gracefully in real ...

Keywords: binary translation, dynamic optimization, dynamic translation, emulation, self-modifying code, speculation

6 XML query processing II: A comprehensive XQuery to SQL translation using dynamic interval encoding
David DeHaan, David Toman, Mariano P. Consens, M. Tamer Özsu
June 2003 **Proceedings of the 2003 ACM SIGMOD international conference on Management of data**
Full text available: pdf(24.20 KB) Additional Information: full citation, abstract, references, citings, index terms

The W3C XQuery language recommendation, based on a hierarchical and ordered document model, supports a wide variety of constructs and use cases. There is a diversity of approaches and strategies for evaluating XQuery expressions, in many cases only dealing with limited subsets of the language. In this paper we describe an implementation approach that handles XQuery with arbitrarily-nested FLWR expressions, element constructors and built-in functions (including structural comparisons). Our propos ...

7 Optimizations and oracle parallelism with dynamic translation
Kemal Ebcioğlu, Erik R. Altman, Michael Gschwind, Sumedh Sathaye
November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.28 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

We describe several optimizations which can be employed in a dynamic binary translation (DBT) system, where low compilation/translation overhead is essential. These optimizations achieve a high degree of ILP, sometimes even surpassing a static compiler employing more sophisticated, and more time-consuming algorithms [9]. We present results in which we employ these optimizations in a dynamic binary translation system capable of computing oracle parallelism.

8 Options for dynamic address translation in COMAs
Xiaogang Qiu, Michel Dubois
April 1998 **ACM SIGARCH Computer Architecture News , Proceedings of the 25th annual international symposium on Computer architecture**, Volume 26 Issue 3
Full text available: pdf(1.37 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

In modern processors, the dynamic translation of virtual addresses to support virtual memory is done before or in parallel with the first-level cache access. As processor technology improves at a rapid pace and the working sets of new applications grow insatiably the latency and bandwidth demands on the TLB (Translation Lookaside Buffer) are getting more and more difficult to meet. The situation is worse in multiprocessor systems, which run larger applications and are plagued by the TLB consiste ...

9 A dynamic binary translation approach to architectural simulation
Harold W. Cain, Kevin M. Lepak, Mikko H. Lipasti
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(830.67 KB) Additional Information: full citation, abstract, index terms

We present the design of a PowerPC-based simulation infrastructure for architectural research. Our infrastructure uses an execution-driven out-of-order processor timing simulator from the SimpleScalar tool set. While porting SimpleScalar to the PowerPC architecture, we would like to remain compatible with other versions of SimpleScalar. We accomplish this by performing dynamic binary translation of the PowerPC instruction set architecture to the SimpleScalar instruction set architecture, and by ...

10 Programmable architectures: Dynamic reconfiguration with binary translation: breaking the ILP barrier with software compatibility
Antonio Carlos S. Beck, Luigi Carro
June 2005 **Proceedings of the 42nd annual conference on Design automation**
Full text available: pdf(411.10 KB) Additional Information: full citation, abstract, references, index terms

In this paper we present the impact of dynamically translating any sequence of instructions into combinational logic. The proposed approach combines a reconfigurable architecture with a binary translation mechanism, being totally transparent for the software designer. Besides ensuring software compatibility, the technique allows porting the same code for different machines tracking technological evolutions. The target processor is a Java machine able to execute Java bytecodes. Experimental resul ...

Keywords: binary translation, java, power consumption, reconfigurable processors

11 Going native: Module-aware translation for real-life desktop applications
Jianhui Li, Peng Zhang, Oma Etzion
June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments**
Full text available: pdf(524.15 KB) Additional Information: full citation, abstract, references, index terms

A dynamic binary translator is a just-in-time compiler that translates source architecture binaries into target architecture binaries on the fly. It enables the fast running of the source architecture binaries on the target architecture. Traditional dynamic binary translators invalidate their translations when a module is unloaded, so later re-loading of the same module will lead to a full retranslation. Moreover, most of the loading and unloading are performed on a few "hot" modules, which caus ...

Keywords: dynamic binary translation, dynamic loaded module, memory management, translation reuse

12 Optimising hot paths in a dynamic binary translator
David Ung, Cristina Cifuentes
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(390.10 KB) Additional Information: full citation, abstract, citings, index terms

In dynamic binary translation, code is translated "on the fly" at run-time, while the user perceives ordinary execution of the program on the target machine. Code fragments that are frequently executed follow the same sequence of flow control over a period of time. These fragments form a hot path and are optimised to improve the overall performance of the program.Multiple hot paths may also exist in programs. A program may choose to execute in one hot path for some time, but later switch to anot ...

Keywords: binary translation, dynamic compilation, dynamic execution, run-time profiling

13 Binary translation and architecture convergence issues for IBM system/390
Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye
May 2000 **Proceedings of the 14th international conference on Supercomputing**
Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, index terms

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

14 Using Event-Based Translation to Support Dynamic Protocol Evolution
Nathan D. Ryan, Alexander L. Wolf
May 2004 **Proceedings of the 26th International Conference on Software Engineering**
Full text available: pdf(160.83 KB) Additional Information: full citation, abstract

All systems built from distributed components involve theuse of one or more protocols for inter-component communication.Whether these protocols are based on a broadlyused "standard" or are specially designed for a particularapplication, they are likely to evolve. The goal of the workdescribed here is to contribute techniques that can supportprotocol evolution. We are concerned not with how or whya protocol might evolve, or even whether that evolution isin some sense correct. Rather, our concern ...

15 Down with Emacs Lisp: dynamic scope analysis

Matthias Neubauer, Michael Sperber
October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming**, Volume 36 Issue 10
Full text available: pdf(218.45 KB) Additional Information: full citation, abstract, references, index terms

It is possible to translate code written in Emacs Lisp or another Lisp dialect which uses dynamic scoping to a more modern programming language with lexical scoping while largely preserving structure and readability of the code. The biggest obstacle to such an idiomatic translation from Emacs Lisp is the translation of dynamic binding into suitable instances of lexical binding: Many binding constructs in real programs in fact exhibit identical behavior under both dynamic and lexical binding. An ...

16 Dynamic typing for distributed programming in polymorphic languages
Dominic Duggan
January 1999 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 21 Issue 1
Full text available: pdf(461.65 KB) Additional Information: full citation, abstract, references, citings, index terms

While static typing is widely accepted as being necessary for secure program execution, dynamic typing is also viewed as being essential in some applications, particularly for distributed programming environments. Dynamics have been proposed as a language construct for dynamic typing, based on experience with languages such as CLU, Cedar/Mesa, and Modula-3. However proposals for incorporating dynamic typing into languages with parametric polymorphism have serious shortcomi ...

Keywords: dynamic typing, marshalling, parametric polmorphism, static typing

17 DAISY: dynamic compilation for 100% architectural compatibility
Kemal Ebcioğlu, Erik R. Altman
May 1997 **ACM SIGARCH Computer Architecture News , Proceedings of the 24th annual international symposium on Computer architecture**, Volume 25 Issue 2
Full text available: pdf(1.97 MB) Additional Information: full citation, abstract, references, citings, index terms

Although VLIW architectures offer the advantages of simplicity of design and high issue rates, a major impediment to their use is that they are not compatible with the existing software base. We describe new simple hardware features for a VLIW machine we call **DAISY** (*Dynamically Architected Instruction Set from Yorktown*). **DAISY** is specifically intended to emulate existing architectures, so that all existing softwa ...

Keywords: binary translation, dynamic compilation, instruction-level parallelism, object code compatible VLIW, superscalar

18 Safe polymorphic type inference for a dynamically typed language: translating Scheme to ML
Fritz Henglein, Jakob Rehof
October 1995 **Proceedings of the seventh international conference on Functional programming languages and computer architecture**
Full text available: pdf(1.33 MB) Additional Information: full citation, references, citings, index terms

19 Optimization and precise exceptions in dynamic compilation
Michael Gschwind, Erik Altman
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(309.41 KB) Additional Information: full citation, abstract, index terms

Maintaining precise exceptions is an important aspect of achieving full compatibility with a legacy architecture. While asynchronous exceptions can be deferred to an appropriate boundary in the code, synchronous exceptions must be taken when they occur. This introduces uncertainty into liveness analysis since processor state that is otherwise dead may be exposed when an exception handler is invoked. Previous systems either had to sacrifice full compatibility to achieve more freedom to perform op ...

20 Word reordering and a dynamic programming beam search algorithm for statistical machine translation
Christoph Tillmann, Hermann Ney
March 2003 **Computational Linguistics**, Volume 29 Issue 1
Additional Information: full citation, abstract, references, index terms

In this article, we describe an efficient beam search algorithm for statistical machine translation based on dynamic programming (DP). The search algorithm uses the translation model presented in Brown et al. (1993). Starting from a DP-based solution to the traveling-salesman problem, we present a novel technique to restrict the possible word reorderings between source and target language in order to achieve an efficient search algorithm. Word reordering restrictions especially useful for the tr ...

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next